SeqBio 2017

Lille

6–7 november 2017

Development of indexing compressed structure for analyzing a collection of similar genomes: application to rice

Clément AGRET[1,2], Gautier SARAH[2,3], Annie CHATEAU[1], Alban MANCHERON[1], Manuel RUIZ[2]

[1] *LIRMM,CNRS and Université de Montpellier, 161 rue Ada, F-34095 Montpellier, France*
[2] *CIRAD, UMR AGAP, Avenue Agropolis, F-34398 Montpellier, France*
[3] *INRA, UMR AGAP, F-34060 Montpellier, France*
***Corresponding author**: agret@lirmm.fr*

Abstract

As the cost of DNA sequencing decreases, the high throughput sequencing technologies become more and more accessible to many laboratories. Consequently, new issues emerge that require new algorithms including tools for indexing and compressing thousands of genomes, as for example the 3000 rice genomes project [1], for which we are particularly interested in.
Genomes can be considered as very large texts on a simple alphabet $\Sigma = \{A, C, G, T\}$, We can refer to indexable dictionary problem which consists in storing a set $S \subseteq \{0, \ldots, i, \ldots, m-1\}$ of an universe $U = n$. $B(n)$ where $B[i] = 1 \iff i \in S$. The indexable dictionary problem support two additionnal operations $rank_s(i)$ and $select_s(i)$ for $s \in \{0, 1\}$. The function $rank_s(i)$ returns the number of elements (s) up to i and $select_s(i)$ returns the position of the i^{th} occurence of s.

The indexation of complete genomes is an important stage in the exploration and understanding of data from living organisms. An efficient index should provide a quick answer to the following questions:
– How many times a given pattern does appear in the genome?
– What are the positions of a given pattern?
– What is the pattern length at the ith position in the genome?

The common way to structure index and compress one genome is to use the Burrows-Wheeler Transform (BWT) [2] with the FM-index [3] on BWT sequences for requests. If you want to index several genomes with one reference genome you may use MuGI [4]. To build MuGI index they store the reference in compact form (4 bits to encode single char), a variant database, one bit vector for each variant and an array kMA keeping information about each k-mers. This is a really interesting approach but it needs to have a reference genome.

We present a structure which proposes a solution to index and compress very repetitive sequences over small alphabet in texts using k-mers. k-mers are factors of length k in the considered sequences. We built a 4^{k_1} array, where $k_1 < k$, and each entry, namely an array, is indexed by a prefix of size k_1 of existing k-mers. In each prefix array we insert a 4^{k_2} bit vector which represents all possible k-mers begining with the considered prefix.
We will use libGkArray [5] to query a large read collections and update our structure. We chose libGkArray instead of JellyFish [6] and KMC (any versions) [7] in main memory.

To build the index, we cut our genomes into k-mers, for each k-mer we split the k-mer into prefix suffix of respective size k_1 and k_2. We call the function $kmer\_to\_int()$ which takes a k-mer and returns its integer value. We then go into the prefix array $PA[kmer\_to\_int(k_1)]$ and we add k_2 to our suffix array. We also add a 1 in the succint structure to G_i $i \in n$ with n the number of genomes as you can see at Fig.1. Given a n for the number of genomes and N for all k-mers in the genome set, we can estimate the time and space complexity as respectively $O(N \log(n))$ and $O(N \times 2k_2 \log(n+N))$. Our structure has to be efficient in memory space and comuting time.

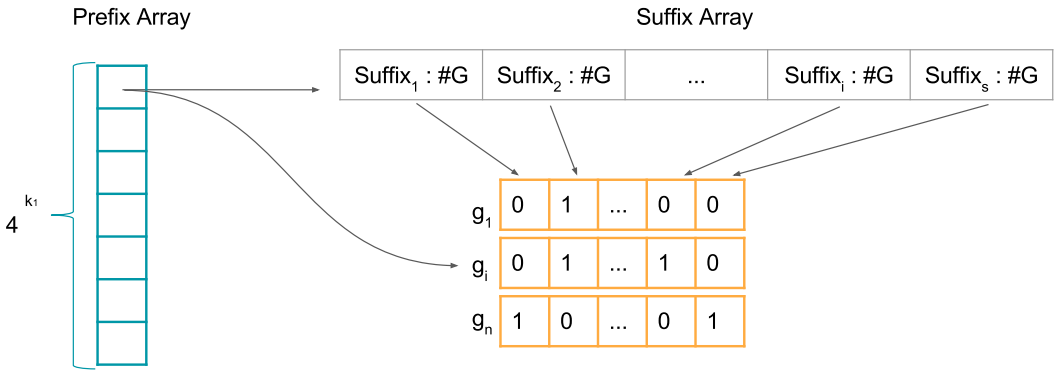

Figure 1. Representation of our structure for n genomes, a k-length $= k$, prefix length: k_1 and a suffix length: $k_2 = s$. We call $\#G$ the number of times we count a given suffix in all genomes.

References

[1] ZK Li and Rutger et al. The 3,000 rice genomes project. *GigaScience*, 3(1):7, 12 2014.

[2] Giovanna Rosone and Marinella Sciortino. *The Burrows-Wheeler Transform between Data Compression and Combinatorics on Words*, pages 353–364. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] Paolo Ferragina and Giovanni Manzini. An experimental study of a compressed index. *Information Sciences*, 135(1):13–28, 2001.

[4] Agnieszka Danek, Sebastian Deorowicz, and Szymon Grabowski. Indexes of large genome collections on a pc. *PLOS ONE*, 9(10):1–12, 10 2014.

[5] Nicolas Philippe, Mikaël Salson, Thierry Lecroq, Martine Léonard, Thérèse Commes, and Eric Rivals. Querying large read collections in main memory: a versatile data structure. *BMC Bioinformatics*, 12(1):242, jun 2011.

[6] G. Marcais and C. Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, mar 2011.

[7] Marek Kokot, Maciej Długosz, and Sebastian Deorowicz. KMC 3: counting and manipulating k-mer statistics. jan 2017.